AFRL-RV-PS-
TR-2013-0106

PERFORMANCE ANALYSIS OF THE ENHANCED BIO-INSPIRED PLANNING ALGORITHM FOR RAPID SITUATION AWARENESS RESPONSE

Yunjun Xu

University of Central Florida
4000 Central Florida Blvd.
Engineering Building 1
Orlando, FL 32765

18 Oct 2013

Final Report

AIR FORCE RESEARCH LABORATORY
Space Vehicles Directorate
3550 Aberdeen Ave SE
AIR FORCE MATERIEL COMMAND
KIRTLAND AIR FORCE BASE, NM 87117-5776

DTIC COPY
NOTICE AND SIGNATURE PAGE

This report is the result of contracted fundamental research deemed exempt from public affairs security and policy review in accordance with SAF/AQR memorandum dated 10 Dec 08 and AFRL/CA policy clarification memorandum dated 16 Jan 09. This report is available to the general public, including foreign nationals. Copies may be obtained from the Defense Technical Information Center (DTIC) (http://www.dtic.mil).

Qualified requestors may obtain copies of this report from the Defense Technical Information Center (DTIC) (http://www.dtic.mil).

AFRL-RV-PS-TR-2013-0106 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

//SIGNED//
KHANH PHAM
Program Manager

//SIGNED//
PAUL HAUSGEN
Technical Advisor, Spacecraft Component Technology Branch

//SIGNED//
BENJAMIN M. COOK, Lt Col, USAF
Deputy Chief, Spacecraft Technology Division
Space Vehicles Directorate

This report is published in the interest of scientific and technical information exchange, and its publication does not constitute the Government's approval or disapproval of its ideas or findings.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	**2. REPORT TYPE**	**3. DATES COVERED** *(From - To)*
18-10-2013	Final Report	4 Apr 2012-4 Oct 2013

4. TITLE AND SUBTITLE

Performance Analysis of the Enhanced Bio-Inspired Planning Algorithm for Rapid Situation Awareness Response

5a. CONTRACT NUMBER
FA9453-12-1-0130

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER
61102F

6. AUTHOR(S)

Yunjun Xu

5d. PROJECT NUMBER
3003

5e. TASK NUMBER
PPM00013649

5f. WORK UNIT NUMBER
EF006375

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

University of Central Florida
4000 Central Florida Blvd.
Engineering Building 1
Orlando, FL 32765

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Air Force Research Laboratory
Space Vehicles Directorate
3550 Aberdeen Avenue SE
Kirtland AFB, NM 87117-5776

10. SPONSOR/MONITOR'S ACRONYM(S)
AFRL/RVSV

11. SPONSOR/MONITOR'S REPORT NUMBER(S)
AFRL-RV-PS-TR-2013-0106

12. DISTRIBUTION / AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

The virtual motion camouflage varying manifold based optimal planning algorithm, inspired by the observation in mating hoverflies, is investigated and the performance is enhanced significantly in terms of the convergence speed and collision avoidance capability. The dimension and time complexities of the innovative algorithm are analyzed. A new strategy based on local pursuit is compared with the virtual motion strategy in constructing the varying manifold. The algorithm is validated in an enhanced low cost robot testbed.

15. SUBJECT TERMS

Bio-inspired trajectory generation, in-situ obstacle avoidance, low-cost LEGO robots, vision-based navigation, macro/micro-level behavior, communication protocols, boundary conditions, formation, stability analysis

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			Khanh Pham
Unclas	Unclas	Unclas	Unlimited	32	**19b. TELEPHONE NUMBER** *(include area code)*

(This page intentionally left blank)

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS AND DISCLAIMER

Acknowledgement

This material is based on research sponsored by Air Force Research Laboratory under agreement number FA9453-12-1-0130. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon.

Disclaimer

The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Air Force Research Laboratory or the U.S. Government.

1 SUMMARY

The meaning of Space Situational Awareness (SSA), which has drawn a lot of attention recently, is more than just the characterization of the space environment and how it will affect space activities [1, 2]. SSA activities can be divided into many possible categories, including but not limited to detecting new space objects and tracking their motion, understanding how the space environment will impact mission behaviors, and determining useful characteristics of identified objects [3]. Interest has also been expressed in using SSA to properly track nearby or distant objects, while being aware of unexpected rendezvous by outside actors [3].

As space becomes more and more crowded, to achieve an effective SSA, not only the capabilities of detecting and tracking objects, but also the capabilities of rapid response and re-planning semi-autonomously or autonomously are crucial. The following are just a few planning problems that are often seen for rapid SSA response: formation keeping, formation reconfiguration, rendezvous and docking, space inspection and assembly without intensive labor, coordinated movement, debris avoidance, and proximity operations considering exhaust plume impingement.

Central to all these planning activities is the capability of computing optimal control commands rapidly while considering nonlinear dynamics, state and control variable constraints, environment limitations, etc. The objectives may be the minimum fuel maneuvers needed to achieve a maximized life time, or the minimum time maneuvers needed to achieve a quick response.

In the performed research, the virtual motion camouflage (VMC) based optimal planning algorithm, inspired by the observation in mating hoverflies, is further investigated and the performance is enhanced significantly in terms of the convergence speed and collision avoidance capability. In addition, the method is compared with the newly developed local pursuit based method. Also the dimension and time complexities of the innovative algorithms are analyzed. The algorithm is validated in an enhanced low cost robots testbed.

2 INTRODUCTION

To date, the majority of local optimal trajectory planning methods belongs to one of the following categories: mathematical programming, heuristic approaches, and hybrid methods. It is worth noting that there are many feasible, real-time methods, such as the A* [4] and the rapidly exploring random tree approaches [5]. The calculus of variations (CoV) together with the Pontryagin's Minimum Principle (PMP) approaches and the Direct Collocation (DC) with nonlinear programming (NLP) approaches are two main methods in the mathematic programming category [6][7]. The CoV + PMP approach works effectively when there is no severe state inequality constraint and a good initial guess is provided. DC + NLP approach can easily incorporate state and control constraints, and its initial guess is not that difficult to obtain. However, the problem dimension of the achieved NLP is normally large and results in a high computational cost. To obtain global optimal solutions, many heuristics and meta-heuristics based methods have been studied, such as evolutionary programming [8] and particle swarming [9]. As mentioned in [10], although these methods work effectively for many different problems, they cannot be rigorously proved and are mostly used off-line due to the high computational cost.

The studied virtual motion camouflage (VMC) and local pursuit (LP) techniques belong to the hybrid category, in which the trajectory planning problem is optimized using a mathematical programming approach in a simultaneously refined manifold. Specifically, in this approach, the path of a vehicle is optimized in a varying subspace or manifold, which is defined by a virtual prey motion and possibly a reference point. The optimization in such a manifold is controlled by a single dimension parameter. The problem dimension in the achieved NLP is small, particularly for the cases with severe constraints and many obstacles. However, just like any other NLP based approaches, a good initial guess is crucial for these methods to converge rapidly, especially when the vehicle is required to navigate in an obstacle-laden environment.

The objectives of the research project are:

O1: The virtual motion camouflage method, inspired by the observations in mating hoverflies, was further investigated for rapid optimal planning problems.

O2: Detailed analyses about the enhanced algorithm were conducted and the performances, such as problem dimension and time complexity, have been investigated. Furthermore the feasibility and flexibility of the algorithm were studied for the cases when the number of obstacles and the number of vehicles in the system are changing.

and

O3: These algorithms were tested in a significantly enhanced low cost robot testbed.

3 METHODS, ASSUMPTIONS, AND PROCEDURES

The research conducted is summarized here: virtual motion camouflage method, initial guess enhancement, local pursuit method, and software update.

3.1 Virtual Motion Camouflage Method

A male hoverfly (**Fig. 1**) uses the motion camouflage to fly along a path connecting the female hoverfly and a background reference point to conceal its motion from the view point of the female one [11].

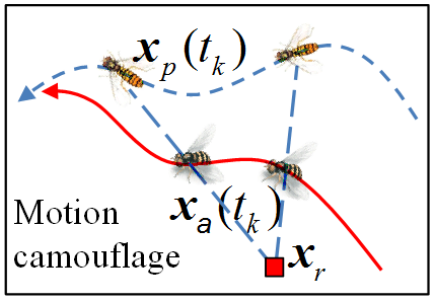

Figure 1. Motion camouflage used by hoverflies

An inherent thread is observed: "the predator only moves along paths in a certain manifold." Inspired by this observation, the simple rule used by the hoverflies can be mimicked to construct varying subspaces (or "manifold") such that the planning problem can be solved rapidly. In the meantime, to avoid potential shortcomings (i.e. the solution optimality depends on the selection of the subspace), the actual vehicle will follow paths that are optimized in an *iteratively refined "manifold"*.

Expanding on these results, the algorithm is outlined in the following steps:

Step 1: Dimension reduction via the motion strategy. The simple rule in the virtual motion strategy is: the "position" state $\boldsymbol{x}$ is confined by the motion of the virtual prey $\boldsymbol{x}_p$, the reference point $\boldsymbol{x}_r$, and the path control parameter (PCP) $v(t)$. The original problem will be converted into the following reduced dimension problem: Given $\boldsymbol{x}_p$, (possibly) the parameters $\boldsymbol{x}_r$, $v(t)$, $\dot{v}(t)$, …, will be designed to minimize the performance index $J_2 = J_2(\boldsymbol{x}_r, v, \dot{v}, \ldots t_f)$ subject to inequality constraints (I.E.Cs.) and equality constraints (E.Cs). It is worth noting that the optimization of the virtual prey motion $\boldsymbol{x}_p$ will be conducted in Step 4 to achieve the optimal solution.

Step 2: Convert the problem achieved in Step 1 into a nonlinear programming problem (NLP) via collocation. Through this discretization, the performance index can be re-written as $J_3 = J_3(\boldsymbol{x}_r, \boldsymbol{v}, t_f)$, in which $\boldsymbol{v} = [v_0, \ldots, v_N]^T$ is the discretized version of the PCP $v(t)$. The derivatives of the PCP variables can be found from $d^n v / dt^n = [2/(t_f - t_0)]^n D^n v$, in which D is the differentiation matrix.

Step 3: Further dimension reduction via the necessary conditions derived from boundary conditions: Certain PCPs, such as the initial PCP (v_0) and final PCP (v_N), can be calculated instead of being optimized. It is worth noting that the necessary conditions are different for different dynamical systems and different boundary conditions.

Step 4: The curve of the virtual prey motion $\boldsymbol{x}_p$ needs be flexible enough so that the solution achieved using the proposed method can be optimal. In the meantime the number of the parameters controlling the prey motion must be small enough such that the computational cost will not be compromised much. In this step, the virtual prey motion will be represented using B-spline curves [12] and the control points $\boldsymbol{P}$ for the B-spine will be optimized. Therefore the final achieved problem is: the performance index can be re-written as $J_4 = J_4(\boldsymbol{x}_r, \boldsymbol{v}, \boldsymbol{P}, t_f)$. The I.E.C.s. and E.C.s. are converted to $\boldsymbol{g}_4(\boldsymbol{x}_r, \boldsymbol{v}, \boldsymbol{P}, t_f) \le 0$ and $\boldsymbol{h}_4(\boldsymbol{x}_r, \boldsymbol{v}, \boldsymbol{P}, t_f) = 0$, respectively.

3.2 Enhanced Initial Guess

In the previous version of the VMC algorithm, a straight line connecting the starting and ending points is used as the initial virtual prey path. For most of the obstacle avoidance cases, the algorithm works well if the reference point in the VMC algorithm is not on this straight line.

However, when the straight line of the virtual prey motion happens to pass through the center of any obstacle, as shown in Fig.2, it becomes challenging for the VMC algorithm (or any other gradient based methods) to rapidly find an optimal or even feasible solution. Therefore, in order to improve the robustness of the VMC method, an alternative approach is investigated in milestone 1 to obtain a good initial virtual prey motion guess.

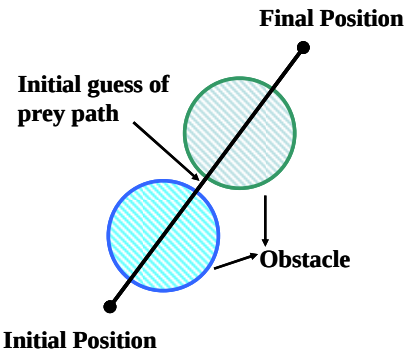

Figure 2. A "bad" initial guess of the virtual prey path

Figure 3 outlines the basic steps involved in finding a good initial guess for the VMC algorithm in finding nonlinear optimal trajectory in obstacle-laden environment.

Step 1: The wavefront algorithm [13] is used to find an obstacle-free corridor, and the virtual prey path in this corridor will generate a trajectory with a high possibility of being obstacle-collision free.

Step 2: Since the path generated by the wavefront method doesn't involve time information and may not be smooth, the result achieved will be regarded as an initial guess and further optimized in a B-spline based optimization. An arbitrary performance index, such as the shortest distance, can be used in the optimization. The obstacle avoidance is regarded as the inequality constraints.

Step 3: The virtual prey found in Step 2 will be used as the initial guess in the VMC method.

Empirically, the initial guess of the PCP variables should be 1. With the enhanced initial guess strategy, the initial selection of a reference point is not as sensitive as the original VMC method. It is more effective if the reference point is placed far away from the actual vehicle's path.

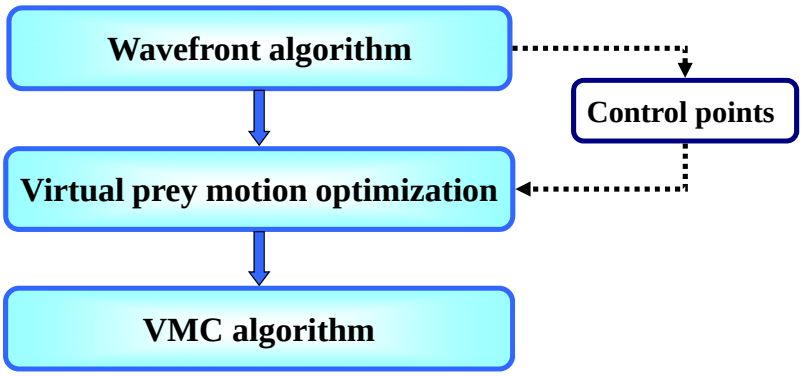

Figure 3. VMC algorithm with enhanced initial guess in obstacle-laden environments

3.3 Local Pursuit Method

The local pursuit (LP) motion strategy is a biological phenomenon found in the behavior patterns of ants. As shown in Fig. 4, an ant points its velocity towards the position of the ant ahead of it to achieve the minimum time performance.

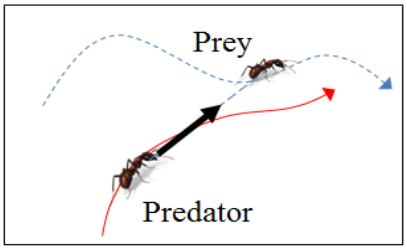

Figure 4. LP strategy

The LP strategy can be described as

$$\dot{\boldsymbol{x}}_a(t) = v\left[\boldsymbol{x}_p(t-\Delta) - \boldsymbol{x}_a(t)\right] \tag{1}$$

in which the position of prey $\boldsymbol{x}_p$ is Δ steps ahead of the predator. v is a scale variable that we call the path control parameter (PCP) so as to be consistent with the VMC method.

Since the prey is a virtual one, the value of Δ will not affect the solution optimality and thus will be regarded as zero. Following the LP motion strategy Eq. (1), the position vector of the Lego robot, $\boldsymbol{x}$, is formulated by the trajectory of the virtual prey, $\boldsymbol{x}_p \triangleq [x_p, y_p]^T$, and PCP, ν, as

$$\dot{\boldsymbol{x}}(t) + v\boldsymbol{x}(t) = v\boldsymbol{x}_p(t) \tag{2}$$

The acceleration derivatives can be derived from Eq. (2), as

$$\dot{\boldsymbol{x}}(t) = \dot{v}(\boldsymbol{x}_p(t) - \boldsymbol{x}(t)) + v(\dot{\boldsymbol{x}}_p(t) - \dot{\boldsymbol{x}}(t)) \tag{3}$$

Then, the control variables V and ω are computed via the dynamic inversion as

$$V = \begin{cases} \dot{x} / \cos(\theta), \cos(\theta) \neq 0 \\ \dot{y} / \sin(\theta), \cos(\theta) = 0 \end{cases} \tag{4}$$

and

$$\omega = (\ddot{y}\dot{x} - \ddot{x}\dot{y}) / (\dot{x}^2 + \dot{y}^2), \ (\dot{x}^2 + \dot{y}^2) \neq 0 \tag{5}$$

respectively. Using a high order discretization method, such as the Legendre-Gauss-Lobatto (LGL) method [14], the position of Lego robot can be further represented as

$$\boldsymbol{\zeta}_j = (D' + \Xi)^{-1} \Xi \boldsymbol{\zeta}_{p,j} \tag{6}$$

in which $\boldsymbol{\zeta}_j = \left[x_{j,0}, ..., x_{j,N} \right]^T$, where $j = 1, 2$ is the j^{th} direction of the position of Lego robot, subscript $0, ..., N$ denotes the discretization node. $\boldsymbol{\zeta}_{p,j} = \left[x_{p,j,0}, ..., x_{p,j,N} \right]^T$ is the discretized version of the virtual prey position. $\Xi = diag\{\boldsymbol{v}\}$ is the diagonal matrix with a proper dimension and $\boldsymbol{v}$ is the discretized version of the SCP variable. D is the differentiation matrix and $D' \triangleq \left[2 / (t_f - t_0) \right] D$.

Similarly, the discretized velocity and acceleration states of the predator can be derived as

$$\dot{\boldsymbol{\zeta}}_j = \left(D' + \Xi \right)^{-1} \left[\dot{\Xi} - \dot{\Xi}(D' + \Xi)^{-1} \Xi \right] \boldsymbol{\zeta}_{p,j} + \left(D' + \Xi \right)^{-1} \Xi \dot{\boldsymbol{\zeta}}_{p,j} \tag{7}$$

and

$$\ddot{\boldsymbol{\zeta}}_j = (D^{'} + \Xi)^{-1} \left\{ \begin{matrix} \ddot{\Xi} - \ddot{\Xi}(D^{'} + \Xi)^{-1}\Xi \\ -2\dot{\Xi}\left(D^{'} + \Xi\right)^{-1}\left[\dot{\Xi} - \dot{\Xi}(D^{'} + \Xi)^{-1}\Xi\right] \end{matrix} \right\} \boldsymbol{\zeta}_{p,j} \\ +2(D^{'} + \Xi)^{-1}\left[\dot{\Xi} - \dot{\Xi}\left(D^{'} + \Xi\right)^{-1}\Xi\right]\dot{\boldsymbol{\zeta}}_{p,j} + (D^{'} + \Xi)^{-1}\Xi\ddot{\boldsymbol{\zeta}}_{p,j} \tag{8}$$

To make the virtual prey motion flexible and enhance the quality of the optimization space, B-spline, which is a method using a small number of control points to define a flexible curve, is chosen to represent the virtual prey motion.

For example, the j^{th} direction of the virtual prey "position" at node k, $x_{p,j,k}, j = 1,2, k = 0,...,N$, can be represented by a nonlinear rational B-spline (NURBS) [12] curve of degree d, as

$$x_{p,j,k} = \sum_{i=0}^{n_{cp}} B_{i,d}\left(t_k\right) P_{j,i}\ ,\ j = 1,2\ ,\ k = 0,...,N \tag{9}$$

where $B_{i,d}\left(t\right), i = 0,...,n_{cp}$ are the d^{th} degree basis functions, $P_{j,i}, i = 0,...,n_{cp}$ are the control points for the j^{th} direction of the virtual prey "position", and $n_{cp} + 1$ is the number of control points. The B-spline representation of the virtual prey motion Eq. (9) can also be written in the vector form as

$$\boldsymbol{\zeta}_{p,j} = B\boldsymbol{P}_j\ ,\ j = 1,...,n_a \tag{10}$$

in which $B = [B_{i,d}\left(t_k\right)]$, $i = 0,...,n_{cp}$, $k = 0,...,N$, and $\boldsymbol{P}_j^T = [P_{j,0},...P_{j,n_{cp}}]$ is the column vector of the control points.

The l^{th} derivative of $x_{p,j,k}$ can be written either in the scalar form as

$$\frac{d^l x_{p,j,k}}{dt^l} = \sum_{m=0}^{n_{cp}} B_{m,d}^{l}\left(t_k\right) P_{j,m},\ j = 1,2, k = 0,...,N \tag{11}$$

or in the matrix form as

$$\boldsymbol{\zeta}_{p,j}^{(l)} = B^{(l)}\boldsymbol{P}_j\ ,\ j = 1,2 \tag{12}$$

in which $B^{(l)} = [B_{m,d}^{(l)}\left(t_k\right)]$ and the superscript (l) represents the l^{th} derivative.

To further reduce the number of optimizable parameters and mitigate the difficulty of the convergence when equality constraints are involved, necessary conditions are derived based on the boundary conditions (BC), which are used to calculate a part of the control points and PCPs. The following BCs are known: the initial and final positions, and the initial velocity.

Thus, after given the first PCP, v_0, the first control point of the virtual prey can be calculated by

$$P_{i,0} = x_{p,i,0} = x_{a,i,0} + \dot{x}_{a,i,0} / v_0 , \ j = 1,2 \tag{13}$$

and the robot's position at all the discretized nodes, except the first and last nodes, will be calculated using,

$$\boldsymbol{\zeta}_{a,i}^{\square} = \left(M_{LP}^{\square}\right)^{+} \left(\Xi B \boldsymbol{P}_i - M_{LP,1}\zeta_{a,i,0} - M_{LP,N+1}\zeta_{a,i,N}\right), \quad i = 1,...,n_a \tag{14}$$

in which $M_{LP} \square (D^{'} + \Xi)$, and $M_{LP,1}$, $M_{LP,N+1}$, and $M_{LP}^{\square}$ are the first, last, and remaining columns of M_{LP}, respectively. $\zeta_{a,i,0}$, $\zeta_{a,i,N}$, and $\boldsymbol{\zeta}_{a,i}^{\square}$ are respectively the first, last, and remaining "position state" variables of the vehicle in the i^{th} direction. $(\ )^{+}$ denotes the pseudo-inverse operation.

Since the BC of the Lego robot has been included in Eq. (13) and Eq. (14) and the dynamic model has been considered in Eq. (4) and Eq. (5), there is no equality constraint in the following formulated nonlinear programming (NLP) problem.

The performance index $J = 0.5\left(t_f - t_0\right)\sum_{i=0}^{N} w_i$ is optimized subject to the inequality constraints $\boldsymbol{g}(\boldsymbol{v}', \boldsymbol{P}', t_f) \le 0$, which includes the limits on the state and control variables and obstacle avoidance. Here $\boldsymbol{v}'$ and $\boldsymbol{P}'$ are the subsets of the parameters in $\boldsymbol{v}$ and $\boldsymbol{P}$ that are optimized.

For convenience, let's define Set S_O to include all the parameters to be optimized, and Set S_C to include the parameters, which are calculated using the necessary conditions derived. The steps involved in the optimization are summarized in Algorithm 1.

Algorithm 1. LP Optimization Algorithm

Steps in the Initialization	Step 1:	Provide initial guesses for the parameters in S_O.
Steps inside the NLP	Step 2:	Calculate the parameters in S_C using the appropriate necessary condition described.
	Step 3:	Construct the virtual prey motion using B-splines.
	Step 4	Compute the state and control variables.
	Step 5:	Evaluate the performance index.
	Step 6:	Evaluate the inequality constraints.
	Step 7:	If the convergence criterion is not satisfied and the maximum number of iterations has not been reached, update the parameters in Set S_O for the next iteration, and go back to Step 2. Otherwise, the optimization is terminated.

3.4 Software and Graphical User Interface (GUI) Updates

The LP based algorithm is programmed and packaged in the Lego robot testbed software (delivered to AFRL-RVSV).

(1) Subroutine: BLP_TRAJ

(2) Objective: Find minimum-time obstacle avoidance optimal path by LP method.

(3) Equation: $\min J = 0.5\left(t_f - t_0\right)\sum_{i=0}^{N} w_i$, such that inequality constraints $\boldsymbol{g}(\boldsymbol{v}', \boldsymbol{P}', t_f) \le 0$, $\boldsymbol{v}'$ is a scalar and $\boldsymbol{P}'$ is a vector.

(4) Syntax

[pos_true,time_true,PATHGEN_CPU,i_count,tf,error_flag]=eval('BLP_TRAJ(struct_robot,i_sorted_points,i_replan_iteration,figs,handles)'): Returns the dimensionalized optimal path, optimal time, central process unit (CPU) calculation time, etc.

(5) Input and Output Arguments and Function Calls. (Table 1)

Table 1. Input and Output Arguments and Function Calls

Variable	**Variable description**	**Dimension**
struct_robot	Information related to robot, and defined in the main document	
i_sorted_points	Pixel locations of two color dots on the Lego robot	1*4
i_replan_iteration(*)	The number of planning horizons	1
figs	Defined in the main document	
handles	Defined in the main document	
Output		
Variable	**Variable description**	**Dimension**
pos_ture	Optimal position (dimensionless) of the Lego robot in the test bed	2*(N+1)
time_true	Time span (dimensionless) discretized by the LGL method	1*(N+1)
pathgen_CPU	The overall CPU time to generate an optimal trajectory by the LP method	1
i_count	Number of tries	1
tf	The minimum time (in dimension) for the Lego robot to move from its initial position to its final position	1
error_flag (**)	Whether or not the initial guess of the prey path is a good one	1

4 RESULTS AND DISCUSSIONS

4.1 Model Used in Simulation and Hardware Validation

The B-spline augmented virtual motion camouflage (BVMC) approach is customized for the two-wheel driven robot with a specific boundary condition (BC). The motion of the robot [15] is governed by the following dynamics model

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos\theta \\ \sin\theta \\ 0 \end{bmatrix} V + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \omega \tag{15}$$

in which $\boldsymbol{r}_a \triangleq [x, y]^T \triangleq [r_{a,1}, r_{a,2}]^T \in \Re^2$ is the midpoint position of the robot, and θ is the heading angle. The position of the robot is confined within the test area as $x_{\min} \le x \le x_{\max}$ and $y_{\min} \le y \le y_{\max}$. The speed V and the turning rate ω are the control variables, which are constrained by $|V| \le V_{\max}$ and $|\omega| \le \omega_{\max}$, respectively. The parameters associated with hardware limitations are found from experiments, which may vary depending on the battery level onboard robots.

The randomly distributed n_{obs} obstacles are assumed to be represented by circles, thus the midpoint positions of the robots need to satisfy $\|\boldsymbol{r}_a - \boldsymbol{r}_{obs,i}\| \ge R_{obs,i} + a_{buf}$, $i = 1, ..., n_{obs}$, where $\boldsymbol{r}_{obs,i}$ is the center of the i^{th} obstacles, and $R_{obs,i}$ is the corresponding radius. a_{buf} is a buffer to take into account the maximum width of the robot from its midpoint.

The planning objective of the robots is to navigate through a series of obstacles from their starting position to chosen ending positions with a minimized final time $J = \int_{t_0}^{t_f} dt$, satisfying the above mentioned constraints. In addition, the robots are required to re-plan their minimum time trajectories, while taking the new additional/pop-up obstacles into account.

The position vector of the robot $\boldsymbol{r}_a$ is optimized within a varying subspace (or "manifold"), which is defined by two main parameters, the trajectory of the virtual prey $\boldsymbol{r}_p \triangleq [x_p, y_p]^T \triangleq [r_{p,1}, r_{p,2}]^T$ and a selected reference point $\boldsymbol{r}_{ref} = [x_{ref}, y_{ref}]^T$ as

$$\boldsymbol{r}_a = \boldsymbol{r}_{ref} + \nu(\boldsymbol{r}_p - \boldsymbol{r}_{ref}) \tag{16}$$

Here a scale variable called the path control parameter (PCP) ν determines how the aggressor behaves inside this varying subspace (or "manifold") using the motion camouflage (MC) rule (Eq. 2). Its velocity and acceleration derivatives are defined as

$$\dot{\boldsymbol{r}}_a = \dot{\nu}(\boldsymbol{r}_p - \boldsymbol{r}_{ref}) + \nu\dot{\boldsymbol{r}}_p \tag{17}$$

and

$$\ddot{\boldsymbol{r}}_a = \ddot{\nu}(\boldsymbol{r}_p - \boldsymbol{r}_{ref}) + \nu\ddot{\boldsymbol{r}}_p + 2\dot{\nu}\dot{\boldsymbol{r}}_p \tag{18}$$

It is worth noting that the reference point is constant but optimized. If $V \neq 0$, the heading angle θ is calculated by $\theta = \tan^{-1}(\dot{y}/\dot{x})$ if $\dot{x} \neq 0$ or $\theta = \sin^{-1}(\dot{y}/V)$ if $\dot{x} = 0$. The speed V and the heading rate ω can be calculated using $V = \sqrt{\dot{x}^2 + \dot{y}^2}$ and $\omega = (\ddot{y}\dot{x} - \ddot{x}\dot{y})/V^2$. If $V = 0$, θ can be any value, and the heading rate can be zero.

There are two simultaneous steps involved in varying the robot trajectory in the problem search space: the subspace (or "manifold") is varied and the actual position of the robot is varied in the subspace. First if the subspace defined by $\boldsymbol{r}_p$ and $\boldsymbol{r}_{ref}$ is fixed, the result obtained may be optimal only in the subspace but not optimal in the space defined in the original problem. To address this issue, the parameters that define the subspace, i.e. the reference point and the virtual prey motion, need to be simultaneously optimized along with the variation of the robot trajectory.

The reference point $\boldsymbol{r}_{ref}$ can be included in the later achieved nonlinear program problem and optimized. The virtual prey motion $\boldsymbol{r}_p$ needs to be varied carefully. In B-spline augmented VMC (BVMC) method, B-spline curves are used to represent the virtual prey motion trajectory $\boldsymbol{r}_p$ as

$$r_{p,i}(t) = \sum_{k=0}^{n_{cp}} B_{k,d}(t) P_{i,k},\ i = 1,2 \tag{19}$$

where $n_{cp}+1$ is the number of the control points $\boldsymbol{P} = [P_{i,k}]$, $k = 0,...,n_{cp}$, $i = 1,2$, determining the shape and direction of the B-spline curves. Unlike polynomials, B-splines have good local control, which allows for more stable curves. $B_{k,d}(t)$, $k = 0,...,n_{cp}$ are the d^{th} degree basis functions.

Second, the position of the robot that is controlled by the PCP variable within the subspace (or "manifold") is discretized to $0,...,N$ nodes using a high order discretization method [14]. The discretized PCP vector is $\boldsymbol{\nu} = [\nu_0, \nu_1, ..., \nu_N]^T \in \Re^{(N+1)\times 1}$, and the k^{th} derivative of the PCP vector is found with the equation

$$\partial^k \boldsymbol{\nu} / dt^k = \left[2/(t_f - t_0)\right]^k D^k \boldsymbol{\nu} \tag{20}$$

in which D is the differentiation matrix [14]. The discretized minimum time cost function is then written as

$$J = 0.5(t_f - t_0)\sum_{i=0}^{N} w_i \tag{21}$$

and the inequality constraint is

$$\boldsymbol{g}\left(\boldsymbol{\nu}, \boldsymbol{P}, \boldsymbol{r}_{ref}, t\right) \leq 0, \quad \boldsymbol{g} \in \Re^{p\times 1} \tag{22}$$

which includes the limits on the state and control variables, obstacle avoidance, and collision avoidance among the robots as described before. The equality constraint $\boldsymbol{h}\left(\boldsymbol{\nu}, \boldsymbol{P}, \boldsymbol{r}_{ref}, t\right) = 0$, $\boldsymbol{h} \in \Re^{q\times 1}$, includes the BC. The dynamic model Eq. (1) has already been considered, thus not included as the equality constraint here.

To further reduce the number of optimizable parameters in the achieved NLP and mitigate the difficulty of the convergence, necessary conditions are derived based on the BC and used to calculate part of the control points and PCPs.

4.2 Robustness of the Enhanced VMC Method

To test the robustness of the enhanced VMC algorithm, many simulation cases with lots of obstacles are conducted. Only six cases are shown in the following pictures. In these simulations, a robot navigating through a test area with randomly generated obstacles is simulated. The cases shown are just six out of the 1000-run Monte Carlo simulation. The CPU time is less than 4.72 seconds when programmed in Matrix Laboratory (MATLAB).

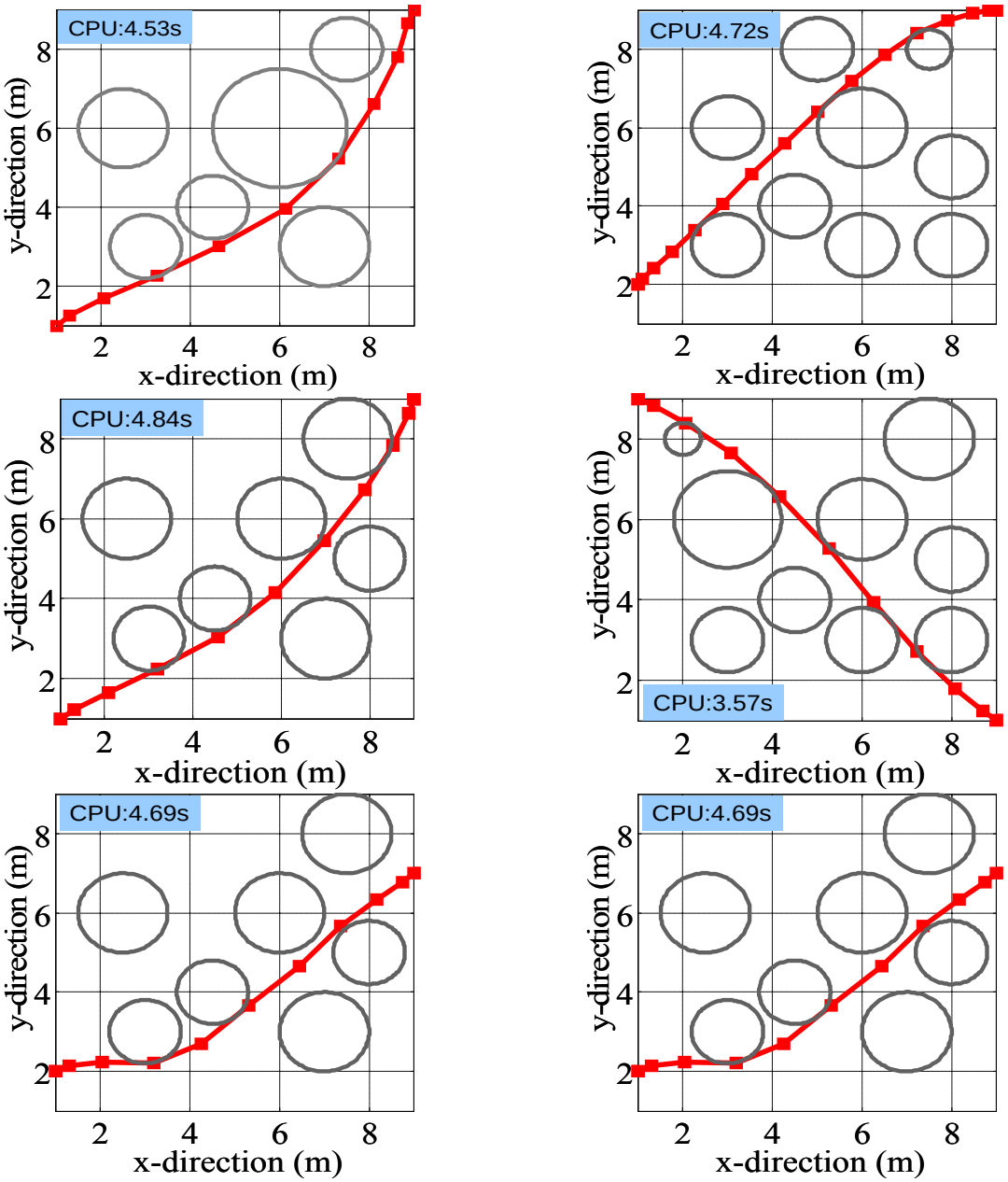

Figure 5. Simulation runs for different settings

4.3 Sensitivity Analysis of the Enhanced VMC Method

In order to study the sensitivity of the parameters in the enhanced VMC method, thousands of Monte Carlo runs are simulated. In the studies, a robot is commanded to move from one corner of the testbed to the other areas in the minimum time, while avoiding obstacles that are randomly generalized. The following tunable parameters are varied: (a) the number of control points of the B-spline curve, (b) the degree of the B-spline curve, (c) the number and the radius of the obstacles, (d) the initial turning angle of the robot; (e) the number of discretization nodes; and (f) the reference point. The following parameters are not varying: (a) the length and width of the testbed (640 pixel x 480 pixel), (b) the initial position and the final position of the robot is

uniformly generated in the corner [0, 20; 0, 20] pixel location and [620, 640; 460, 480] pixel location of the testbed, respectively; (c) 500 Monte Carlo runs are performed for each tunable parameter case; and (d) the initial PCPs are set to be one.

The parameter is regarded as sensitive if the change of this parameter will produce significant change in the successful rate. For example, the number of control points is sensitive as shown in Table 2. When the number of control points is 5, 29 out of 500 runs fail; while if the number of control points is set to be 8, only 7 out of 500 runs (1.4%) fail.

It can be seen from the following table that only the degree of B-spline is less sensitive. Therefore, it can be concluded that it is important to tune the parameters properly in order to achieve very robust results.

Table 2. Sensitivity Analysis

Tunable Parameters	**Sensitive or Not**	**Examples**			
		Case 1	**# Failed/500**	**Case 2**	**# Failed/500**
# of Control Points	Sensitive	5	29/500	8	7/500
Degree of B-Spline	Less sensitive	4	2/500	6	5/500
# of Discre-tization Points	Sensitive	10	20/500	15	6/500
Reference Point	Sensitive	-[1800,1800] pixel	6/500	-[18000,18000] pixel	39/500
Initial Angle	Sensitive	$[0\ 90]^o$	25/500	$[40\ 45]^o$	6/500
# of Obstacles	Sensitive	[1 10]	2/500	[30 40]	51/500
Radius of Obstacles	Sensitive	[5 15] pixel	6/500	[20 30] pixel	125/500

In this section, ten hardware experiments are collected and compared. All the programs are written in the Matlab (R2010b) with the following toolboxes: Acquisition, Image Processing, RWTH-Mindstorms Nxt, and Optimization. All computations were performed on the same laptop with a frequency of 2GHz and random access memory (RAM) of 6GB.

4.4 Experiments Comparison between VMC and LP

Experiment Settings:

These ten experiments are categorized in two groups. In the first group, the whole optimization is completed in one planning horizon, while in the second group, two planning horizons are used to handle the unexpected pop-up obstacles.

In each test, the Lego robot will move from a randomly generated initial position to a randomly placed final position using both the LP and VMC methods. Four or five obstacles with the same radius will be placed randomly on the test bed. The maximum translational speed of the Lego robot is measured as $V_{\max} = 15\,cm/s$, while the maximum rotational speed is $\omega_{\max} = 16\,rad/s$.

Performance Index and Success Rate Comparison:

The performance index (PI) and CPU time of the 1st group are shown in Table 3.

Table 3. The performance index and CPU time of the 1st group

Group 1		LP	VMC
Test 1	CPU Time	3.69	8.08
	PI	29.24	31.56
Test 2	CPU Time	2.16	3.44
	PI	30.29	30.23
Test 3	CPU Time	4.81	32.28
	PI	38.41	39.54
Test 4	CPU Time	4.72	5.44
	PI	39.22	39.54
Test 5	CPU Time	3.11	5.10
	PI	36.17	37.18
Test 6	CPU Time	5.03	9.51
	PI	33.13	34.07

For each test, the minimum performance index among these two methods is regarded as the "best" solution. If the performance index from the other method is within 5% difference of the best solution, the solution found using this method is regarded as an "optimal" one; otherwise it will be regarded as a "feasible" solution only.

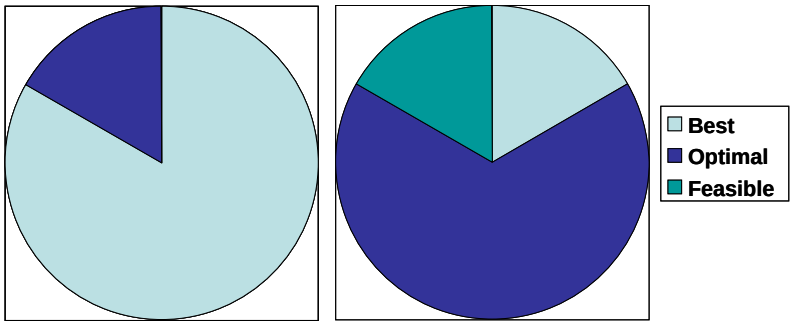

Figure 6. Use Case 1 for solutions obtained using the LP method and the VMC method

From Table 2 and Fig. 6, the following observations are obvious: (i) Both the LP and VMC methods can help Lego robot find minimum-time path successfully, the largest difference between the PIs is only 7%. (ii) The LP method has much higher "best" solution rate in finding

the optimal solutions as compared with that of the VMC method. (iii) The CPU time needed in the VMC method is in the range of 3-9s, while the LP method has a much smaller CPU time, which is in the range of 2-5s.

It is worth mentioning that to record the experiment results, some unnecessary CPU time is wasted in exporting some information during the optimization.

The performance index (PI) and CPU time of the 2nd groups are shown in Table 4 and the comparison of solutions is shown in Fig. 7. Observations similar to the first group can be made for the second group. It is worth noting that the performance index is changed during the two planning horizons because the distance to the desired position is reduced.

Table 4. The performance index and CPU time of the 2nd group

Group 2		LP	VMC	LP	VMC
		1st planning		2nd planning	
Test 1	CPU Time	3.29	9.23	2.67	8.42
	PI	31.28	31.83	20.41	22.05
Test 2	CPU Time	4.25	6.33	2.41	4.15
	PI	30.45	29.79	19.23	18.39
Test 3	CPU Time	4.05	6.35	2.78	6.81
	PI	29.40	30.78	18.80	20.64
Test 4	CPU Time	3.97	6.45	2.07	4.69
	PI	34.75	36.13	21.82	22.38

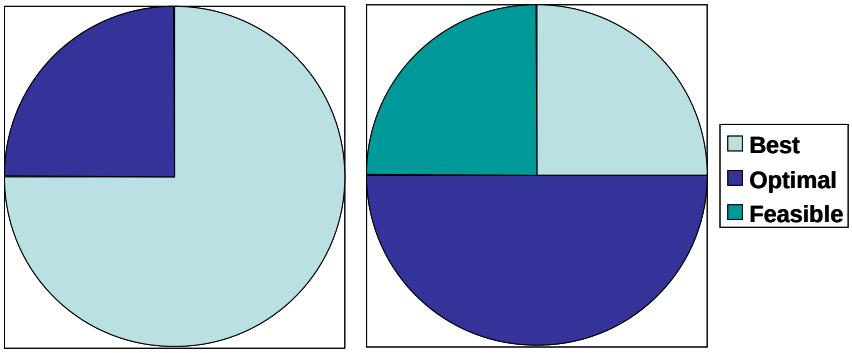

Figure 7. Use Case 2 for solutions obtained using the LP method and the VMC method

Optimal Trajectories:

Two tests (Test 2 and Test 4) from the 1st group are selected and shown in Fig. 8 and Fig. 9, respectively. It can be seen from Fig. 8 and Table 2 that the optimal trajectories (represented by the red lines) and the minimum time to arrive at the desired position calculated by the VMC method and the LP method are approximately the same, and Lego robot can track the optimal trajectory (represented by the blue, dotted line) very well using the low level tracking controller. Seen from Fig. 9 and Table 3, the optimal trajectory generated by the LP method is smoother

than the one generated by the VMC method, and the minimum time calculated by the LP method is smaller than that of the VMC method.

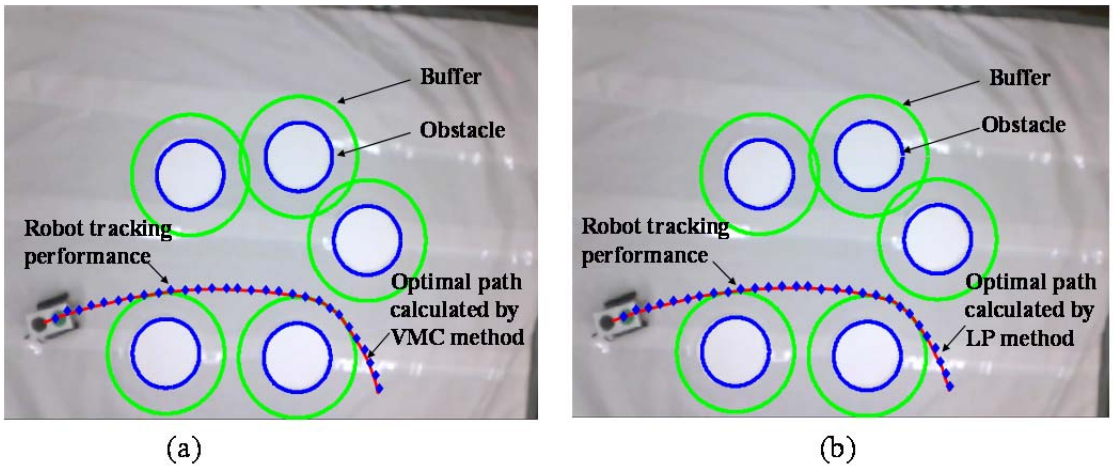

Figure 8. The optimal trajectory generated in test 2 of the 1st group: (a) VMC method; (b) LP method

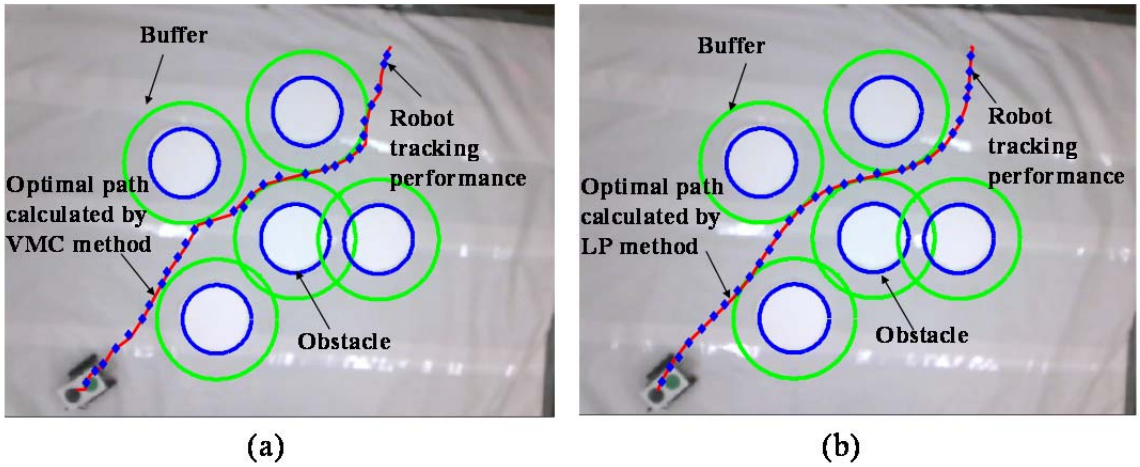

Figure 9. The optimal trajectory generated in test 4 of the 1st group: (a) VMC method; (b) LP method

One result from the 2nd group is shown in the Fig. 10. Obstacles 1-3 are known *a priori*, while obstacles 4-5 are popped up obstacles. As shown in Fig. 6, once the optimal trajectories are computed using either the VMC method or the LP method, the Lego robot will follow the generated path. When new obstacles appear, the Lego robot will stop and wait for a new optimal trajectory to be generated and then follow the re-planned trajectory to reach the desired final position. It can be seen that the planned path in the first horizon passes through obstacle 4 (unknown), while the re-planned trajectory corrects the trajectory in the second horizon.

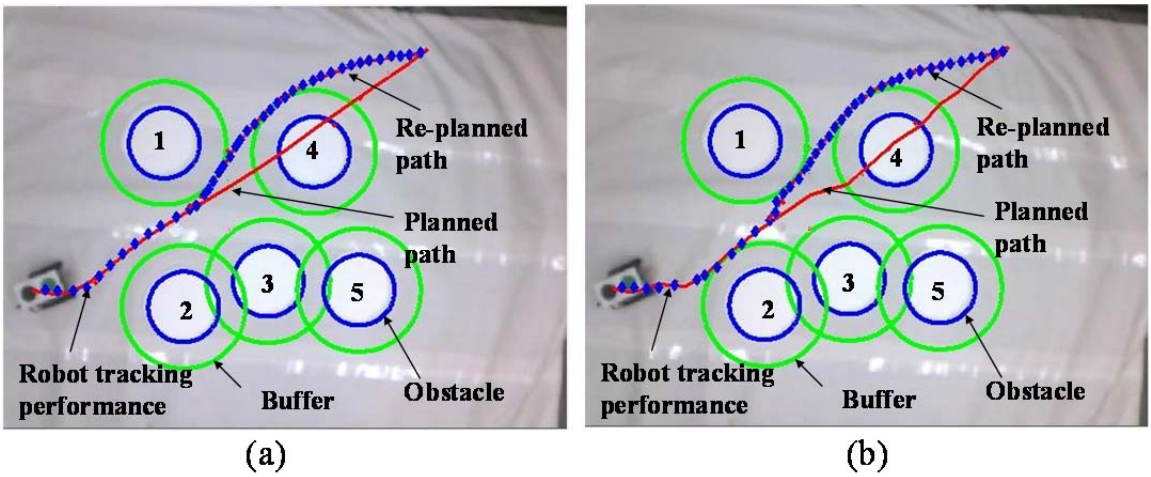

Figure 10. The optimal trajectories generated in test 3 of the 1st group: (a) VMC method; (b) LP method

5 CONCLUSIONS

In this research, the VMC trajectory optimization method is enhanced. The performance of the VMC method is compared with a new strategy (LP) based method. It is concluded that the LP method can achieve a better performance than the VMC method in terms of convergence rate, optimization speed, and solution optimality. New perturbation methods are proposed to enhance the initial guess.

The following publications have been achieved:

- N., Li, Y. Xu, K. Pham, "Micro Air Vehicle's 3D Trajectory Planning and Parametric Estimation," 2013 AIAA GNC, Boston, MA, August 18-22, 2013.

- A journal version is under preparation.

The following software packages have been delivered to AFRL-RV.

- Software Version 1.0 for 2D Robot Trajectory Planning in Obstacle-Laden Environment (In report 1)

- Software Version 2.0 for 2D Robot Trajectory Planning in Obstacle-Laden Environment (Monte Carlo Simulation Version) (in Report 2)

- Software Version 3.0 for 2D Robot Trajectory Planning in Obstacle-Laden Environment (Monte Carlo Simulation Version) (in Report 3)

- Ten Experiment Videos are submitted (in Report 3).

6 FUTURE WORK

Some potential research and development improvements include: (i) The LP method needs to be further enhanced and used in decentralized cooperative trajectory planning problems. (ii) System identification or estimation methods can be used online to quantify the coefficients of the nonlinear dynamic model and a receding horizon framework can be used to update the command generation. Stability needs to be proven for such a system. (iii) A helicopter platform needs to be incorporated into the testbed to provide real-time information about the test area from a moving platform. (iv) Communication issues such as delay is one of the next steps to be studied.

REFERENCES

[1] Teehan, R. F., *Responsive Space Situation Awareness in 2020*, Center for Strategy and Technology, Blue Horizons Paper, Air War College. April 2007.

[2] Abbot, R. I., and Wallace, T. P., "Decision Support in Space Situational Awareness," *Lincoln Laboratory Journal*, Vol. 16, No. 2, 2007, pp. 297-335.

[3] Erwin, R.S., P., Jayaweera, S. K., and Hussein, I., "Dynamic Sensor Tasking for Space Situational Awareness," *American Control Conference*, Baltimore, Maryland, 2010.

[4] Trovato, K. I., and Dorst, L., "Differential A*," *IEEE Transaction on Knowledge and Data Engineering*, Vol. 14, No. 6, 2002, pp. 1218-1229.

[5] Yang, K., and Sukkarieh, S., "3D Smooth Path Planning for a UAV in Cluttered Natural Environments," *IEEE International Conference on Intelligent Robots and Systems*, Nice, France, September 22-26, 2008, pp. 794-800.

[6] Goerzen, C., Kong, Z., and Mettler, B., "A Survey of Motion Planning Algorithms from the Perspective of Autonomous UAV Guidance," *Journal of Intelligent and Robotic Systems*, Vol. 57, No. 1-4, 2010, pp. 65-100.

[7] Betts, J., Practical methods for optimal control using nonlinear programming, Society for Industrial and Applied Mathematics, Philadelphia, 2001.

[8] Rathbun, D., and Capozzi, B., "Evolutionary Approaches to Path Planning through Uncertain Environments," *AIAA's 1st Technical Conference and Workshop on Unmanned Aerospace Vehicles, Systems, Technologies, and Operations*, Portsmouth, VA, May 20-23, 2002, AIAA 2002-3455.

[9] Roberge, V., Tarbouchi, M., and Labote, G., "Comparison of Parallel Genetic Algorithm and Particle Swarm Optimization for Real-time UAV Path Planning," *IEEE Transaction on Industrial Informations*, Vol. 9, No. 1, 2013, pp. 132-141.

[10] Y. Xu, and G. Basset, "Sequential Virtual Motion Camouflage Method for Nonlinear Constrained Optimal Trajectory Control," *Automatica*, Vol. 48, No. 7, 2012, pp. 1273-1285.

[11] Srinivasan, M. V., and Davey, M., "Strategies for Active Camouflage Motion," *Proceedings of the Royal Society of London Biological Sciences*, Vol. 259, 1995, pp. 19-25.

[12] Piegl, L., and Tiller, W., *The NURBS Book: Second Edition*, Springer-Verlag, New York, 1997.

[13] Batavia, P. H., and Nourbakhsh, I., "Path Planning for the Cye Personal Robot," *Proceedings of IEEE/RSJ Internation Conference on Intelligent Robots and Systems*, Pittsburgh, PA, USA, October 31-November 5, 2000, pp. 15-20.

[14] Fahroo, F., and Ross, I. M., "Costate Estimation by a Legendre Pseudospectral Method," *Journal of Guidance, Control, and Dynamics*, Vol. 24, No. 2, 2001, pp. 270-275.

[15] Laumond, J. P., Sekhavat, S., & Lamiraux, F. (1998). Guidelines in nonholonomic motion planning for mobile robots. *Lecture Notes in Control and Information Sciences*, LNCIS 229, New York: Springer-Verlag, pp. 1-44.

LIST OF ACRONYMS, ABBREVIATIONS, AND SYMBOLS

ACRONYM	Description
BC	Boundary Condition
BVMC	B-Spline Augmented Virtual Motion Camouflage
CoV	Calculus of Variations
CPU	Central Process Unit
DC	Direct Collocation
E.C.	Equality Constraints
GUI	Graphical User Interface
I.E.C.	Inequality Constraints
NLP	Nonlinear Programming
LGL	Legendre-Gauss-Lobatto
LP	Local pursuit
MATLAB	Matrix Laboratory
MC	Motion Camouflage
NURBS	Non-uniform Rational B-Spline
PI	Performance Index
PMP	Pontryagin's Minimum Principle
PCP	Path Control Parameter
RAM	Random Access Memory
SSA	Space Situational Awareness
VMC	Virtual Motion Camouflage

DISTRIBUTION LIST

DTIC/OCP
8725 John J. Kingman Rd, Suite 0944
Ft Belvoir, VA 22060-6218 1 cy

AFRL/RVIL
Kirtland AFB, NM 87117-5776 2 cys

Official Record Copy
AFRL/RVSV/Khanh Pham 1 cy

(This page intentionally left blank)